Docket Number: POU9-2001-0026US1

Inventor: BERNARD IULO

Title METHOD AND SYSTEM FOR MANAGING COMPUTER PERFORMANCE

APPLICATION FOR

UNITED STATES LETTERS PATENT

"Express Mail" Mailing Label No.: ET089965423US

Date of Deposit: November 1, 2001

I hereby certify that this paper is being deposited with the United States Postal Service as "Express Mail Post Office to Addressee" service under 37 CRF 1.10 on the date indicated above and is addressed to Box Patent Application, Assistant Commissioner for Patents, Washington, D.C. 20231.

Name: Ann S. Lund

Signature: Ann S. Lund

METHOD AND SYSTEM FOR MANAGING COMPUTER PERFORMANCE

BACKGROUND

[0001] This invention relates generally to managing computer system performance, and more particularly, the present invention relates to a method and system for tracking and logging system environmental and task information during successful and uninterrupted system operations for use in diagnosing future system problems.

[0002] During their execution, tasks, jobs, or programs running on various computer operating systems commonly experience errors. A minor error, such as where a task completes successfully although a warning appears which requires review, is a frequently occurring error and is one that, fortunately for the user, causes little or no damage. A severe error, such as where a task unexpectedly terminates without completing (also referred to as an abend), can cause significant harm to a running application.

[0003] Historically, diagnosing and correcting these problems involved examining the error data provided at the time of the failure, determining what was not working correctly, and eventually providing a fix or a recommendation that allowed the task to work properly. Most operating systems on the market today include methods of capturing an image of system activity at the time of the failure (referred to as a dump), a chronological mapping of system activity before and during a failure (referred to as a trace), or a log of system and task errors and environmental data also around the time of the problem (error logs).

[0004] This data is effective for many problems a task may encounter. But as the complexity of operating systems continues to increase and the dynamics of the computing environment continue to change, coupled with the variety and number of independent vendors and programmers introducing changes to operating systems, the frequency and complexity of problems associated with these events continues to challenge system analysts. Meeting these challenges requires more than the traditional environmental techniques, but also an understanding of the full scope of this expanding environment.

[0005] Currently, when questions about system changes and past successful task processing arise, the data available is usually based upon the recollection of an individual system programmer. This often results in error-prone analyses and incomplete data. Changes can occur in so many ways and so frequently that it is becoming impractical to rely on the memory of an individual who may not even be aware of all of the changes taking place.

[0006] What is needed, therefore, is a method and system that will address this new and expanding set of complex system problems in a disciplined and precise manner.

BRIEF SUMMARY

[0007] An exemplary embodiment of the invention relates to a method and system for managing computer performance. The system comprises a server connected to a network link and a data storage device in communication with the server. The data storage device provides a repository for a system activity database which stores activity records and environmental records created by a system monitor application, as well as an auxiliary database which stores historical records transferred from the system activity database. The system also includes a client system in communication with the server, a user interface for the system monitor program that is executable by the server, and a system activity analysis program executable by the server. Upon execution of the system activity analysis program, the server searches the system activity and auxiliary databases, retrieves data relating to a request for analysis of a task, and presents the data to a user on the client system.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] Referring now to the drawings wherein like elements are numbered alike in the several FIGURES:

[0009] FIG. 1 is an exemplary system diagram of a computer network upon which the system monitor program and system activity analysis program are implemented in a preferred embodiment; and

[0010] FIG. 2 is an exemplary flowchart illustrating the process of resolving system errors via the execution of the system activity analysis program tool.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0011] In an exemplary embodiment, the system monitor program and system activity analysis program are implemented in a network environment as shown in FIG. 1. System 100 includes an enterprise 102 including a server 104, client systems 106, a data storage device 108, and a network link 110 for enabling entities of enterprise 102 communicate with one another.

[0012] Enterprise 102 may be a commercial establishment such as a business enterprise. Server 104 may be operating various software applications for enabling entities of enterprise 102 to communicate with each other as well as perform various enterprise level tasks. Software applications may include enterprise resource planning software, web server and applications server software, groupware, email software, database management software, and other tools commonly used in a business environment. Server 104 is also executing system monitor software for capturing system activities and related environmental data.

[0013] Server 104 is also executing the system activity analysis program tool of the invention. The system activity analysis program tool is invoked by an authorized user, preferably a system administrator, analyst, or programmer for enterprise 102, when traditional methods of debugging a problem detected on a system device have failed. These scenarios may include cases where a routinely-executed task, having a previous record of successful execution, has recently failed.

[0014] Data storage device 108 stores databases of records generated by the system monitor program and provides a summary of activities which have occurred over a

specified period of time. Databases housed in data storage device 108 include a system activity database 112 and an auxiliary database 114. System activity database 112 stores activity records 116 and environmental records 118 generated and provided by the system monitor tool. Auxiliary database 114 stores historical data periodically transferred from system activity database 112 for future reference. Reports records 120 are also available via the system activity analysis program tool for reference and/or trend analyses.

[0015] The system monitor program captures specified elements about the successful execution of a job or task. It also captures key environmental information including items such as system resources requested and used (e.g., amount of storage, critical locks/latches held, dataset/files, execution time) and saves this information in system activity database 112. When a task is started, the monitor captures the task name, start time, data sets/files allocated, initial storage in an activity record. As the task executes, the system monitor captures other execution variables such as system locks/latches used, storage used, and status of any sub- tasks performed. This function of the system monitor continues as long as the task is active. As required, more information may be added to the activity record.

[0016] When the task completes, the system monitor will complete the tracking of the task by including in the activity record the end time for the task, locks/latches freed, storage used, data sets/files updated, and the final status on the completion of the task. Upon completion of the task, the resulting activity record is written to system activity database 112.

[0017] The activity record is used to track the details relating to the execution of a specific task. The format of activity record 116 is illustrated below.

Activity record identifier
Activity flags
Unique Task Name
Initial Storage Allocated
Name of Initial Program Called
Pointer to List of Additional Storage Requests
Count of Number of Additional Storage Requests
Pointer to List Data Sets/Files used
Count of Number of Data Sets/Files
Pointer to List of Latches/Locks Used
Count of Number of Latches/Locks Used
Execution Start Time
Execution End Time

[0018] Definitions of fields provided in activity records 116 are provided below.

[0019] Activity Record Identifier. X'AAAAAAAA0001'

[0020] Activity Flags. Special task indicator flags include a positive indicator if the task has completed successfully and a negative indicator if the task has not completed successfully.

[0021] Unique Task Name. A unique name (up to 1024 bytes) that is assigned to the task by an authorized system user.

[0022] Initial Storage Allocated. The number of initial bytes of storage, either set or defaulted, that is assigned by the installation.

[0023] Name of Initial Program Called. The name of the first program that gets invoked for the task.

[0024] Pointer to a List of Additional Storage Requests. A pointer to another table which includes all additional storage requests, time made, amount of storage requested and indicators to note if the storage has been freed.

[0025] Count of Number of Additional Storage Requests. A counter to keep track of the number of additional storage requests made.

[0026] Pointer to a List of Data Sets/Files Used. A pointer to another table which includes all data set names or file names used by the program.

[0027] Count of Number of Data Sets/Files Used. A counter to keep track of the number of data sets/files used.

[0028] Pointer to a List of Locks/Latches Used. A pointer to another table which includes all locks and latches used by the program as well as an indicator for when the latch/lock is freed.

[0029] Count of Number of Latches/Locks Used. A counter to keep track of the number of latches/locks used.

[0030] Execution start time. System time stamp at task start.

[0031] Execution end time. System time stamp at task end.

[0032] Environmental records 118 may be provided for every unique system or application program available on enterprise system's 102 operating system. These records 118 are used to monitor system change activity and are input to the analysis program. Many operating systems today provide a variety of installation tools that gather much of this data. The system monitor program includes an interface that can be invoked to monitor these environmental changes. The format of environmental record 118 is shown below.

Environmental Record Identifier
Environmental Flags
Program or Product Name
Time Stamp for Change, Addition, or Deletion
Name of OEM Vendor for Product
Phone Number for Contact at OEM Vendor

[0033] Field definitions for environmental record 118 are described below.

[0034] Environmental record identifier. X'EEEEEEEE0001'

[0035] Environmental flags. Task indicator flags including indicators that classify a task as a software program, a hardware product, an initial entry, a change to the program or product, a removal of a program or a product, a change in the name of an OEM vendor, or a change in the contact phone number.

[0036] Time stamp. A system time stamp indicating when the event described in the environmental record took effect.

[0037] Name of OEM (original equipment manufacturer) vendor for product. The name of the OEM manufacturer or developer of the product in this environment record.

[0038] Phone number for contact at OEM vendor. Phone number of technical person responsible for servicing this program.

[0039] System activity database 112 is the repository for all activity and environmental records, 116 and 118 respectively, generated during the execution of the system monitor tool. Preferably, this repository has an allocation sufficient to contain at least one full day of system operation. Further, all records 116 and 118 stored in system activity database 112 will preferably be off loaded to auxiliary database 114 on a daily basis for potential later use.

[0040] The system activity analysis program tool may be initiated by an authorized user of enterprise system 102 when a problem has been discovered concerning a job or task running on an entity of enterprise system 102 for which traditional debugging techniques have failed. FIG. 2 illustrates the process of resolving a system error detected by a system user utilizing the system activity analysis program tool. At step 202, a system error has been encountered by a user of one of client systems 106. Traditional debugging techniques have been exhausted and so the system administrator invokes the system activity analysis program application via one of client systems 106 at step 204. The system activity analysis program tool prompts the administrator to enter a task name relating to the task being analyzed at step 206. Once entered, the system activity analysis program tool searches system activity database 112 and auxiliary database 114 for all previous executions of the identified task at step 208. Details of previous executions found in the activity record for the task are then displayed and/or printed out for the administrator at step 210. Details are provided for each previous execution of the task up to and including the time that the problem occurred.

[0041] The system activity analysis program tool next locates the activity record for the failing execution at step 212 and presents it to the administrator for review. This record includes the sequence of task activities which reveal how far the task executed and what it was doing when the problem occurred. From this information, the administrator may be able to determine the root cause of the problem (step 214). If so, the administrator exits

the program and takes corrective action according to the information received. If no cause is apparent to the administrator (step 214), the administrator may continue by requesting a listing of previous environmental change records created between the last successful execution of the program up to the time of the failure at step 216. The tool retrieves these records from system activity database 112 and presents them to the administrator at client system 106 for review. From this information, the administrator may be able to determine if the root cause is related to a system or program change. Again, the administrator may have enough information to determine the root cause of the failure (step 218). If so, the administrator may exit the program and take appropriate corrective action. If the cause is not apparent, however, the administrator may request that the system activity analysis program tool run a contention analysis report which will map out locks/latches obtained and also identify locks/latches held by other tasks in system activity database 112 at step 220. If still no cause is determined (step 222), the administrator now has more detailed information directed to activities occurring at and near the time of the failure with which to perform traditional problem analysis such as system dumps and traces at step 224. The administrator may also utilize the system activity record for the task generated by the system monitor tool as a task-level trace that can be validated against any successful task runs at step 226. If this is the first execution of a task on this system, obviously there is no successful activity record to trace against, however, the 'task-tracing' included in the failing system activity record and the system activity analysis program contention analysis report would still provide much more detail than is currently available by traditional analysis tools and techniques.

[0042] The system activity analysis program tool provides a mechanism for tracking the successful execution of various jobs and tasks routinely running in a system. It captures specified environmental information including system resources requested and used, and saves this information in a database, providing a task profile which is continually updated each time the same job or task is executed successfully. The tool also logs specified system environmental factors at various points during the ongoing running of a computer

system, including system changes made such as new products installed and changes to the operating system. An analysis routine provided by the tool may also be invoked when a system problem is encountered and provide information to the person diagnosing the problem including profiles for any jobs/tasks running on the system during the time of the problem. The invention may help identify abnormal variations in job or task characteristics from normal execution, potential contention and storage issues, and also a history of any changes in the system environment since the last point these tasks ran successfully.

[0043] By tracking system activities and environmental changes, the tool can save system administrators valuable time and resources that would otherwise be necessary in investigating these problems. Detailed account management profiles of the operating environment provide useful information in assisting the administrator in resolving these problems. Further environmental records provide a facility available to identify by program either a direct contact that can work on resolving a problem or the number for a support center to handle the problem.

[0044] As described above, the present invention can be embodied in the form of computer- implemented processes and apparatuses for practicing those processes. The present invention can also be embodied in the form of computer program code containing instructions embodied in tangible media, such as floppy diskettes, CD-ROMs, hard drives, or any other computer- readable storage medium, wherein, when the computer program code is loaded into and executed by a computer, the computer becomes an apparatus for practicing the invention. The present invention can also be embodied in the form of computer program code, for example, whether stored in a storage medium, loaded into and/or executed by a computer, or transmitted over some transmission medium, such as over electrical wiring or cabling, through fiber optics, or via electromagnetic radiation, wherein, when the computer program code is loaded into and executed by a computer, the computer becomes an apparatus for practicing the invention. When implemented on a general-purpose microprocessor, the computer program code segments configure the microprocessor to create specific logic circuits.

[0045] While preferred embodiments have been shown and described, various modifications and substitutions may be made thereto without departing from the spirit and scope of the invention. Accordingly, it is to be understood that the present invention has been described by way of illustration and not limitation.